* Keyword

|  |  |  |  |
| --- | --- | --- | --- |
| False | def | if | raise |
| None | del | import | return |
| True | elif | in | try |
| and | else | is | while |
| as | except | lambda | with |
| assert | finally | nonlocal | yield |
| break | for | not |  |
| class | from | or |  |
| continue | global | pass |  |

## 

All the built in function

['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'ZeroDivisionError', '\_', ' build\_class ', '\_\_debug ', ' doc\_\_', ' import ', '\_\_loader ', '\_\_name ', '\_\_package ', '\_\_spec\_\_', 'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'exec', 'exit', 'filter', 'float',

'format', 'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input',

'int', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'map', 'max',

'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'quit',

'range', 'repr', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod',

'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']

## help( builtins )

Help on built-in module builtins:

NAME

builtins - Built-in functions, exceptions, and other objects.

DESCRIPTION

Noteworthy: None is the `nil' object; Ellipsis represents `...' in slices.

CLASSES

object

BaseException

Exception

ArithmeticError FloatingPointError OverflowError ZeroDivisionEr

### Help on built-in function input in module builtins:

input(...)

input([prompt]) -> string

Read a string from standard input. The trailing newline is stripped.

If the user hits EOF (Unix: Ctl-D, Windows: Ctl-Z+Return), raise EOFError. On Unix, GNU readline is used if enabled. The prompt string, if given, is printed without a trailing newline before reading.

(END)

#### Help on built-in function print in module builtins:

print(...)

print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default. Optional keyword arguments:

file: a file-like object (stream); defaults to the current sys.stdout. sep: string inserted between values, default a space.

end: string appended after the last value, default a newline. flush: whether to forcibly flush the stream.

(END)

Help on built-in function dir in module builtins: dir(...)

dir([object]) -> list of strings

If called without an argument, return the names in the current scope.

Else, return an alphabetized list of names comprising (some of) the attributes of the given object, and of attributes reachable from it.

If the object supplies a method named dir , it will be used; otherwise the default dir() logic is used and returns:

for a module object: the module's attributes.

for a class object: its attributes, and recursively the attributes of its bases.

for any other object: its attributes, its class's attributes, and recursively the attributes of its class's base classes.

* Void function – return 'None'
* Fruitful function – return something which is not 'None'
* print() - a void function
* input() - a fruitful function
* # - anywhere
* Docstring – script level, class level, and function level – must be at the top of each level
* len(string) - > number of characters
* len(list) → number of items
* len(dictionary) → number of key-value items
* len(set) → number of collection items
* len(int) → error
* len(file\_object) → error

\*\*\*\*\*\*\*Create data objects

* number = 10
* name = 'hello'
* s = '' ←empty string
* l = [ ] ← empty list
* d = {} ← empty dictionary
* t = set() ← empty set

Numeric and string objects

* a = 10
* b = 20
* c = a + b → c = 30
* s = 'hello'
* t = 'world'
* u = s + t → u = 'hello world'

Integer and string objects

* n = 20
* e = '='
* w = e \* 20 → '===================='
* e \* 12.5 → type error

String methods

* s = 'this is a short string'
* s[0:1] → 't'
* s[0:4] → 'this'
* s[10:15].upper() → 'SHORT'
* Etc.
* my\_list = []
* my\_list.append('b') → ['b']
* my\_list += ['a'] → ['b','a']
* my\_list.sort() → ['a','b']
* Other list methods:
  + 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort

grades = {}

* grades['ops435']= 'A' → {'ops435':'A'}
* grades.pop('ops435') → 'A'
* grades = {'ops435':'B','int420':'A'}
* grades.popitem() → ['ops435','B']
* Other dictionary methods:
  + 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values’

import

* import os → can call os.listdir()
* import argparse as arg
* from os import listdir → can call listdir()
* from os import \* → danger! Not recommended

File objects

* f = open('testfile.txt','r') → file object for reading
* contents = f.read() → the entire file is assigned to the string object named contents
* Contents = f.readlines() → the entire file is assigned to the list object named contents. Each item in the list ended with a new line character.
* f.close() → close the opened file.
* f = open('data.txt','w') → f object for writing
* Other file methods and attributes:
  + 'buffer', 'close', 'closed', 'detach', 'encoding', 'errors', 'fileno', 'flush', 'isatty', 'line\_buffering', 'mode', 'name', 'newlines', 'read', 'readable', 'readline', 'readlines', 'seek', 'seekable', 'tell', 'truncate', 'writable', 'write', 'writelines'

age = 10

name = 'Tommy'

Try:

print(age + name)

Except TypeError:

print(“Both age and name must be integer.”)